FIR Filter Design: Hilbert Transformer's Example

Karan Batra

November 25, 2008

Contents

List of Figures

Abstract

A Finite Impulse Response (FIR) filter can be created by applying simple shifting and truncation on the analog prototype filter. Although shifting is a simple operation, proper care must be taken while applying the discrete-time shifting property, which ceases to follow for a fractional shift. Here we take the example of a Hilbert Transformer to confirm this fact. The MATLAB scripts are included in the appendix.

1 Introduction

The ideal frequency response of a digital Hilbert transformer is:

$$H(\Omega) = \begin{cases} -j & \Omega > 0 \\ 0 & \Omega = 0 \\ j & \Omega < 0 \end{cases} = \begin{cases} -j\ sgn(\Omega) & \Omega \neq 0 \\ 0 & \Omega = 0 \end{cases} \tag{1}$$

The relation used to compute the inverse DTFT of $H(\Omega)$ is given as:

$$h_n = \frac{1}{2\pi} \int_{2\pi} H(\Omega) e^{j\Omega} d\Omega \tag{2}$$

Plugging $H(\Omega)$ from (1) into (2):

$$h_n = \frac{1}{2\pi} \left(\int_{-\pi}^{0} j\ e^{j\Omega} d\Omega + \int_{0}^{\pi} -j\ e^{j\Omega} d\Omega \right) = \frac{2}{\pi n}$$

Shifting and Truncation are necessary steps for converting this non-causal filter to a realizable FIR filter. In short,

- Shifting helps capture most of the power in the frequency spectrum
- Truncation enables one to implement this filter on a digital piece of hardware viz. a microcontroller or a DSP

Following the DTFT shifting property,

$$h[n - N_s] = H(\Omega) e^{-j\Omega N_s} \quad \forall \quad N \in Z$$

As specified in the lab handout, the ideal response is shifted by $N_s = (N-1)/2$. Here is the catch, $N_s \notin Z$ for an even value of N. That will be a fractional shift, and DTFT shifting property doesn't applies in this case. Hence, we can only apply the shifting property for odd values of N. The next section derives $H_{shift,N}$ in a more robust fashion, and thus works for any value of N (even or odd).

2 A More General Approach

Plugging $H_{shift,N}[\Omega]$ from Lab#7 handout into (2),

$\Rightarrow h_{shift,N} = \frac{1}{2\pi} \left[\int_{-\pi}^{0} j e^{-j\Omega(N-1)/2} e^{jn\Omega} d\Omega + \int_{0}^{\pi} -j e^{-j\Omega(N-1)/2} e^{jn\Omega} d\Omega \right]$

$\Rightarrow \frac{j}{2\pi} \left[\int_{-\pi}^{0} e^{j\Omega(n-(N-1)/2)} d\Omega - \int_{0}^{\pi} e^{-j\Omega(n-(N-1)/2)} d\Omega \right]$

$\Rightarrow \left(\frac{j}{2\pi(n-(N-1)/2)} \right) \left[e^{j\Omega(n-(N-1)/2)}]_{-\pi}^{0} + e^{j\Omega(n-(N-1)/2)}]_{0}^{\pi} \right]$

$\Rightarrow \left(\frac{1}{2\pi(n-(N-1)/2)} \right) \left[1 - e^{-j\pi(n-(N-1)/2)} - e^{j\pi(n-(N-1)/2)} + 1 \right]$

Equation (3) is the most general equation, which applies to both odd and even values of N.

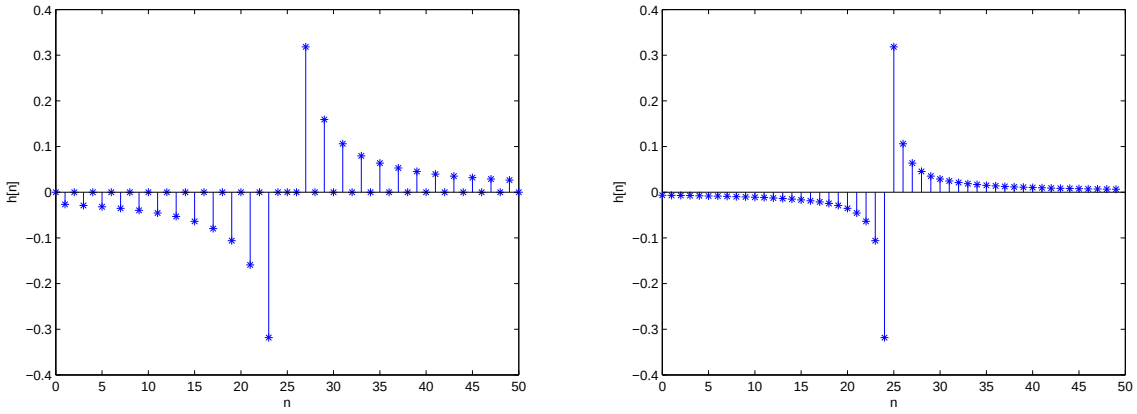
$$h_{shift,N} = \left(\frac{1}{2\pi(n-(N-1)/2)} \right) \left(1 - cos\left(\left(n - \frac{N-1}{2} \right) \pi \right) \right) \tag{3}$$

Looking at the previous result in light of N being odd/even:

Care must be taken while reading 'n' and 'N' because we are trying to analyze both of them in odd/even perspective. For N as a odd integer we get a simplified expression as in (4).

$$h_{shift,odd-N} = \begin{cases} \frac{2}{\pi\{n-(N-1)/2\}} & n-odd \\ 0 & n-even \end{cases} \tag{4}$$

However, for N being an even integer, there is no such simplification possible (Use (3)) and as it turns out, every other element is not zero as it is in the odd-N case. An odd length Hilbert Transformer will therefore be computationally more efficient than an even length one. The following figure depicts the case for length 51 and 50.



(a) Length-51 Hilbert Transformer

(b) Length-50 Hilbert Transformer

Figure 1: FIR Filter Design: The Hilbert Transformer

A Computing the Filter Coefficients

Listing 1: MATLAB Script for Computing Hilbert FIR Filter-Coefficients

```
colorgrey = [211,211,211]/255;
N= 51;
n = 0:N-1;
h = 2./(pi*(n-(N-1)/2)).*(rem(n,2)~=0);
h(26) = 0;
stem(n, h, '*'), xlabel('n'), ylabel('h[n]')

N = 50;
n = 0:N-1;
h = (1 ./ (2*pi*(n-(N-1)/2))) .* (1-cos(pi*(n-(N-1)/2)));
figure, stem(n, h, '*'), xlabel('n'), ylabel('h[n]')
```

B Writing a Header File

Listing 2: MATLAB Script for Writing a Header File

```
% Assuming h is in the workspace
fid = fopen('header.h','wt');
fprintf(fid, 'float h[]={');
fprintf(fid, '%12g,',h); %12g specifies the precision
fprintf(fid, '};');
fprintf(fid,'\n');
fclose(fid);
```